buffer 68 were somehow directly rendered in region 249 without the processing performed by compositor 76, the image displayed by display device 83 should appear to be magnified as shown in FIG. 12.

The compositor 76 is configured to blend the graphical data in frame buffers 66-69 and to composite or combine the blended data and the graphical data from frame buffer 65 such that the screen 247 displays the image shown by FIG. 10. In particular, the compositor 76 blends into a single pixel each set of four pixels that were previously super-sampled from the same pixel by pipeline 56. This blended pixel should have a color value that is a weighted average or a blend of the color values of the four super-sampled pixels. Furthermore, the compositor 76 also blends into a single pixel each set of four pixels that were previously super-sampled from the same pixel by pipeline 58. This blended pixel should have a color value that is a weighted average or a blend of the color values of the four super-sampled pixels. Thus, the object 284 should appear in anti-aliased form within portions 266 and 268, as depicted in FIG. 10.

The super-sampling performed by pipelines 56-59 should improve the quality of the image displayed by display device 83. Furthermore, since each pipeline 56-59 is responsible for rendering only a portion of the image displayed by display device 83, similar to the optimization mode, the speed at which a super-sampled image is rendered to display device 83 can be maximized.

**Jitter Mode**

Referring to FIG. 3, the operation and interaction of the client 52, pipelines 55-59, and the compositor 76 will now be described in more detail while each of the pipelines 55-59 is operating in the jitter mode. In the jitter mode, each pipeline 56-59 is responsible for rendering the graphical data defining the entire 3D image to be displayed within region 249.

Thus, each pipeline 56-59 refrains from discarding portions of the graphical data based on inputs received from slave controller 261, as described hereinabove for the optimization and super-sampling modes. Instead, each pipeline 56-59 renders the graphical data for each portion of the image visible within the entire region 249.

However, each pipeline 56-59 adds a small offset to the coordinates of each pixel rendered by the pipeline 56-59. The offset applied to the pixel coordinates is preferably different for each different pipeline 56-59. The different offsets applied by the different pipelines 56-59 can be randomly generated by each pipeline 56-59 and/or can be pre-programmed into each pipeline 56-59. After the pipelines 56-59 have applied the offsets to the pixel coordinates and have rendered to frame buffers 66-69, respectively, the compositor 76 combines the graphical representation defined by the data in each frame buffer 66-69 into a single representation that is rendered to the display device 83 for displaying. In combining the graphical representations, the compositor 76 averages or blends the color values at the same pixel locations in frame buffers 66-69 into a single color value for the same pixel location in the final graphical representation that is to be rendered to the display device 83.

The aforementioned process of averaging multiple graphical representations of the same image should produce an image that has been jitter enhanced. The drawback to enhancing the image quality in this way is that each pipeline 56-59 renders the entire image to be displayed within region 249 instead of just a portion of such image as described in the optimization and super-sampling modes. Thus, the amount of time required to render the same image may be greater for the jitter mode as opposed to the optimization and super-sampling modes. However, as compared to conventional systems 15 and 41, the amount of time required for the system 50 to render a jitter enhanced image should be significantly less than the amount of time required for either of the conventional systems 15 or 41 to produce the same jitter enhanced image.

In this regard, in performing jitter enhancing in a conventional system 15 or 41, a single pipeline 23 or 36-39 usually renders the graphical data defining an image multiple times to enable jitter enhancement to occur. Each time the pipeline 23 or 36-39 renders the graphical data, the pipeline 23 or 36-39 applies a different offset. However, in the illustrated environment, a different offset is applied to the same graphical data via multiple pipelines 56-59. Therefore, to achieve the same level of jitter enhancement of an image, it is not necessary for each pipeline 56-59 of system 50 to render the graphical data defining the image the same number of times as the single conventional pipeline 23 or 36-39. Thus, the system 50 should be able to render an jitter enhanced image faster than conventional systems 15 and 41.

To better illustrate the operation of the system 50 in the jitter mode, assume that the application 17 issues a command to display the 3D object 284 depicted in FIG. 10. In this example, graphical data defining the object is transmitted from the client 52 to the master pipeline 55. The master pipeline 55 transmits this graphical data to each of the slave pipelines 56-59. Each of the slave pipelines 56-59 renders the graphical data defining the 3D object 284 to frame buffers 66-69, respectively. In rendering the graphical data, each pipeline 56-59 adds a small offset to each set of coordinate values within the graphical data defining the object 284. The offset added by each pipeline 56-59 is preferably different and small enough such that the graphical representations of the object, as defined by frame buffers 66-69, would substantially but not exactly overlay one another, if each of these representations were displayed by the same display device 83.

As an example, pipeline 56 may add the value of .1 to each coordinate rendered by the pipeline 56, and pipeline 57 may add the value of .2 to each coordinate rendered by the pipeline 56. Further, pipeline 58 may add the value of 0 to each coordinate rendered by the pipeline 58, and the pipeline 59 may add the value of -.2 to each coordinate rendered by the